



البرمجة الشيئية

Object Oriented Programming (with Java)

ITGS211

المحاضرة الثالثة

الفصل الدراسي: ربيع 2017

جمل الادخال والاخراج Input & Output Statements

- عادةً الـ keyboard تعتبر جهاز الادخال القياسي *standard input* device الذي تُستقبل منه البيانات، وأما المعلومات فتُرسل لجهاز العرض وهو غالباً الشاشة باستخدام Class مخزن في *standard Java library*.
- *Java classes* المخزنة في *standard Java library* يتم الوصول إليها باستخدام *Java Applications Programming Interface* أو مايشار إليه اختصاراً بـ *Java API*.
- في المثال السابق:

```
System.out.println(message);
```

- تم استخدام *System class* من *standard Java library* وهذا الـ *Class* يحوي *methods* تؤدي وظائف في *system level*.
- **out** هو *object* من *System class* ويحوي الوظيفتين *print* و *println* اللتان تؤديان مهمة ارسال المعلومات إلى جهاز العرض *output device*.

3

```
public class HelloWorld {
    // this is a HelloWorld program
    public static void main(String[] args) {
        // TODO code application logic here
        String message = "Hello World";
        System.out.println(message);
    }
}
```

ut - HelloWorld (run) %

```
run:
Hello World
```

4

- وفي مثالنا السابق سيتم ارسال قيمة ما بين القوسين وهو المتغير message الذي يحوي السلسلة **Hello World** فيكون ناتج تنفيذ هذا السطر هو



Hello World

- لاحظ فيما سبق من أمثلة فإن:
- **جملة الطباعة print statement**: تستخدم في عملية اخراج البيانات والمعلومات على وتأخذ الشكل التالي:
- `System.out.print(Exp);`
 - حيث: Exp يمكن أن يكون ثابت عددي أو قيمة لمتغير أو تعبير حسابي أو منطقي.
 - أحياناً قد يحتاج المبرمج طباعة أكثر من تعبير في نفس جملة الطباعة كما يلي:
- `System.out.print(Exp1 + Exp2 + ...);`
 - لاحظ الفصل بين التعبيرات يكون بالمؤثر (+).
- **جملة الطباعة println statement**: لها نفس شكل جملة print اعلاه إلا أنها تسبب في انتقال مؤشر الطباعة لسطر جديد قبل عملية الطباعة التالية.

5

أمثلة

- `System.out.print("OO-Programming");`
- يطبع OO-Programming أي النص مابين علامتي التنصيص كما هو.
- `System.out.print(40+60);`
- يقوم بجمع العددين ثم يطبع الناتج وهو 100.
- **`int A=40 , B=60 , c=A+B;`**
 - `System.out.println(A+B);`
 - يقوم بجمع قيمتي المتغيرين A و B أولاً ثم طباعة الناتج وهو 100. ثم يجعل مؤشر الطباعة ينزل سطر جديد.
 - `System.out.print(c);`
 - يطبع قيمة المتغير c وهي 100 ، في سطر جديد.

6

- `System.out.print("A+B=" + 40+60);`
يطبع النص مابين علامتي التنصيص كما هو `A+B=` متبوعاً بـ 40 ثم 60 وكأنهما سلسلة، فيكون ناتج الطباعة كالتالي `A+B= 4060` ⇐ ⇐
- `System.out.print("A+B= " + A+B);`
يطبع النص مابين علامتي التنصيص كما هو `A+B=` متبوعاً بقيمة المتغير `A` وهي 40 ثم قيمة المتغير `B` وهي 60 كالتالي `A+B= 4060` ⇐ ⇐
- `System.out.print("A+B= " + (A+B));`
يطبع النص مابين علامتي التنصيص كما هو `A+B=` متبوعاً بحاصل جمع قيمتي المتغيرين `A` و `B`، فيكون ناتج الطباعة `A+B= 100` ⇐ ⇐
- `System.out.print("A+B= " + "A"+"B");`
يطبع النص مابين علامتي التنصيص كما هو `A+B=` متبوعاً بالحرفين `A` و `B`، فيكون ناتج الطباعة `A+B= AB` ⇐ ⇐

7

جملة القراءة

- **Scanner Class**: يستخدم لقراءة البيانات من المستخدم عن طريق لوحة المفاتيح. وهو موجود في `java.util` لذلك يجب استخدام الجملة التالية في أعلى البرنامج.
- ```
import java.util.Scanner;
```
- الكائن **Scanner** يعمل مع `System.in` ولتكوينه نستخدم:
  - `Scanner input = new Scanner(System.in);`  
هذا السطر يكتب مرة واحدة ويبدل على أن عملية إدخال ستحدث خلال البرنامج
  - `input.nextLine ();`  
هذه الجملة تدل على قراءة `input` من المستخدم `user` ويمكن أن يكتب أكثر من مرة خلال البرنامج وفي هذه الحالة يتم قراءة `string` فقط.

8

## جملّة القراءة

- `input.nextInt () ;`

• وفي هذه الحالة يتم قراءة عدد صحيح `int` فقط .

- `input.nextFloat () ;`

• وفي هذه الحالة يتم قراءة عدد حقيقي `float` فقط .

9

```
import java.util.Scanner;

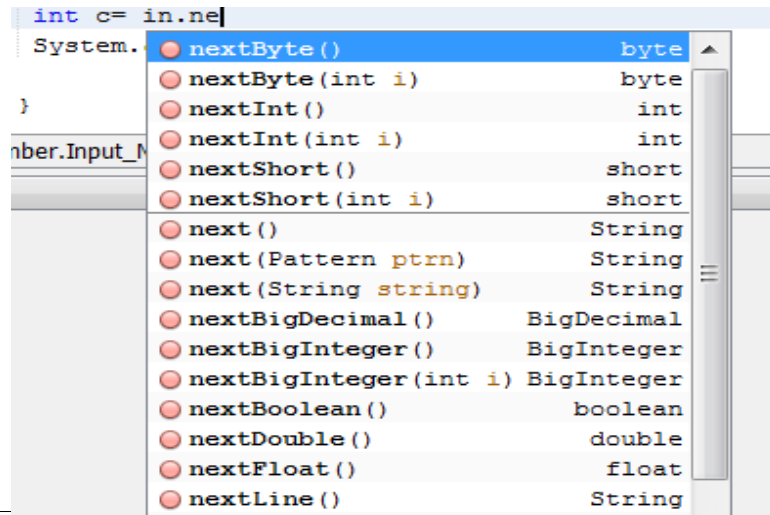
public class Input_Number {

 public static void main(String[] args) {
 Scanner in= new Scanner(System.in);
 System.out.println("Enter int number");
 int c= in.nextInt();
 System.out.println("C= " + c);
 }
}
```

10

## جملة القراءة

لاحظ تم استخدام `in.nextInt()` في المثال السابق لادخال قيمة عددية صحيحة.



11

## مثال 1

اكتب برنامج لحساب مساحة مستطيل طول ضلعه (x) وعرضه (y) علماً بأن: مساحة المستطيل تساوي الطول \* العرض.  
المساحة (Z) = (y \* x)

```
package area;
import java.util.Scanner;
public class Area {
 public static void main(String[] args) {

 Scanner input = new Scanner (System.in);
 System.out.println("Enter height");
 int y =input.nextInt();
 System.out.println("Enter width");
 int x=input.nextInt();
 int z=x*y;
 System.out.println("Area="+z);
 }
}
```

12

## مثال 2

أكتب برنامج لحساب متوسط ثلاثة أعداد ؟

```
import java.util.Scanner;
public class Average {

 public static void main(String[] args) {
 Scanner input = new Scanner(System.in);
 int x=input.nextInt();
 int y=input.nextInt();
 int z=input.nextInt();
 float s=x+y+z;
 float a=s/3;
 System.out.print("Average=");
 System.out.println(a);
 }
}
```

Output - Average (run) x

```
run:
12
14
5
Average=10.333333
BUILD SUCCESSFUL (t
```

13

## مثال 3

إذا كان لديك مبلغ مالي (100 دينار) وتريد شراء سلعة بقيمة (40 دينار) لكل قطعة، كم عدد القطع التي يمكن شرائها بهذا المبلغ؟ وكم سيتبقى لديك من المبلغ الحالي؟ أكتب برنامج بلغة Java لحل هذه المسألة.

```
import java.util.Scanner;
public class Modul {

 public static void main(String[] args) {
 Scanner input = new Scanner(System.in);
 int m = input.nextInt();
 int p = input.nextInt();
 int n = m/p;
 int r= m%p;
 System.out.println(n);
 System.out.println(r);
 }
}
```

Output - modul (run) x

```
run:
100
40
2
20
BUILD SUCCESSFUL
```

14

## أنواع المعالجة Process Types

15

المقصود بأنواع المعالجة هو كيفية تنفيذ جهاز حاسوب لجمل البرنامج وهناك أربع طرق لذلك وهي :

1. **المعالجة المتتابعة Sequential Process (step by step)**
2. **المعالجة الاختيارية Selection Process (if , if else , Switch case)**
3. **المعالجة التكرارية Repetition Process (for , do while , do until)**
4. **المعالجة العشوائية Randm Process (Go to)**

16



Operation 1

Operation 2

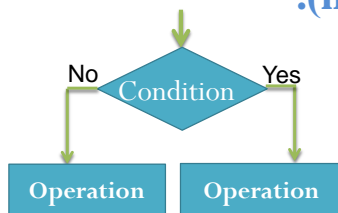
Operation 3

Operation \_n

### المعالجة التتابعية (step by step).

- يقوم مبدأ هذا النوع على أن يبدأ تنفيذ الجمل جملة بعد جملة وبالترتيب من أعلى إلى أسفل، فينفذ الجملة الأولى ومنها ينتقل الى الجملة الثانية وهكذا حتى آخر جملة.

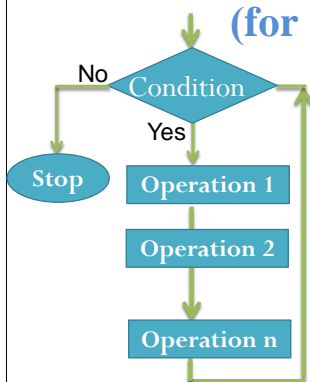
### المعالجة الاختيارية (if , if else , switch).



- في هذا النوع يقوم باختيار جملة أو مجموعة جمل ليقيم بتنفيذها حسب شرط معين Condition، وجملة أو مجموعة جمل لا يقوم بتنفيذها. ولتطبيق هذا النوع من المعالجة يجب استخدام إحدى الجمل البرمجية التالية: ( if , if else , switch )

17

### المعالجة التكرارية (for , do\_while,do\_until)



- في هذا النوع من المعالجة يتم تكرار تنفيذ جملة أو مجموعة جمل أكثر من مرة حسب شرط معين ويتم ذلك باستخدام إحدى الجمل التكرارية.

### المعالجة العشوائية (Go to)

- في هذا النوع من المعالجة يتم تنفيذ الجمل البرمجية بشكل عشوائي فمثلاً ينفذ الجملة الأولى ثم ينتقل إلى الجملة السابعة ثم ينفذ الجملة الرابعة ثم الثانية وهكذا. ويتم ذلك باستخدام جملة (Go to).

18

## أنواع الشروط Condition Types

- تقسم الشروط إلى نوعين:

### 1- الشرط البسيط Simple Condition

لتكوينه نستخدم أدوات المقارنة Comparison operator

### 2- الشرط المركب Complex Condition

لتكوينه نحتاج لاستخدام أدوات المقارنة Comparison Operator بالإضافة إلى الأدوات المنطقية Logical Operator.

19

## جملة الشرط if statement

وتأخذ الصيغة العامة التالية:

```

→ If (condition)
{
 Statement_1;
 :
 Statement_n;
}
→ next statement

```

معنى هذه العبارة أنه إذا كان الشرط condition الذي تقوم الجملة (if) باختباره صحيحاً فيتم تنفيذ الجملة/الجمل المحصورة بين القوسين، وفي حال عدم صحة الشرط فلا يتم تنفيذ تلك الجمل بل يقفز لتنفيذ بقية جمل البرنامج (next statement)، أي يتخطى جملة if وما تحويه بين قوسيه.

20

## جملة الشرط if statement

• في حالة تنفيذ جملة واحدة فقط بعد جملة (if) فإنه يمكن الاستغناء عن الأقواس وفي هذه الحالة تنتهي جملة الشرط بالفاصلة المنقوطة (;) كما يلي :

```
If (condition)
 Statement-1;
```

**ملاحظة:** حتى وإن كانت جملة if تحتوي على أكثر من سطر ستبقى جملة واحدة one statement، كما في المثال التالي:

```
if (average > 95)
 grade = 'A';
```

تكافئ وظيفياً التالي:

```
if(average > 95) grade = 'A';
```

21

## جملة الشرط if statement

**لاحظ أن:** في حال عدم وجود قوسي المجموعة فإنه سيعتبر الجملة الأولى فقط true statement

```
if (Condition)
 statement1;
 statement2;
 statement3;
 next statement;
```

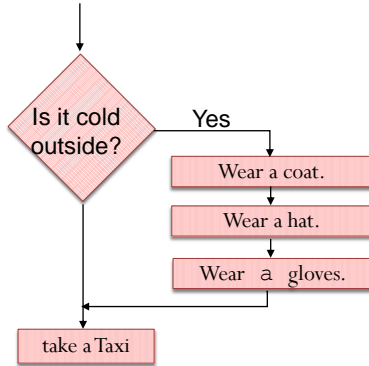
22

## مثال توضيحي

```

if (coldOutside)
{
 wearCoat();
 wearHat();
 wearGloves();
}
takeTaxi();

```

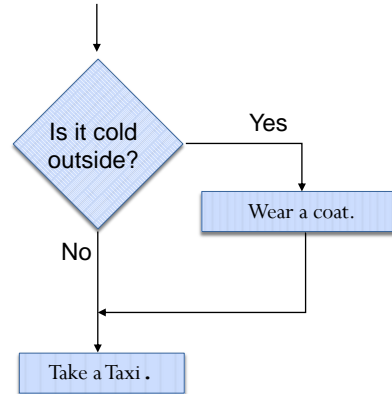


23

```

if (coldOutside)
 wearCoat();
 takeTaxi();

```



## جملة الشرط if else statement

```

if(Condition)

```

```

{
 statement1;
 statement2;
}

```

```

else

```

```

{
 statement_n1;
 statement_n2;
}

```

تأخذ الجملة if else الصيغة العامة:

يعتبر هذا النوع من الجمل الشرطية  
امتداد للجمل الشرطية if statement  
حيث تعبر هذه الجملة عن انه إذا تحقق  
الشرط *Condition* فننفذ *الجملتين*

```
statement1;
```

```
statement2;
```

إذا لم يتحقق الشرط فننفذ *الجملتين*

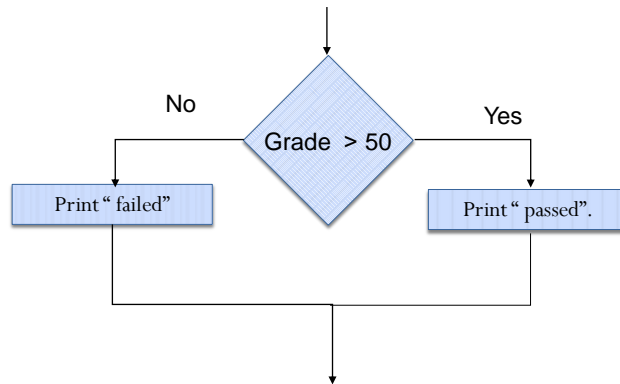
```
statement_n1;
```

```
statement_n2;
```

24

## if-else Statement Flowcharts

يمكن استخدام جملة if else عند التحقق من الشرط وإذا تحقق يتم تنفيذ جملة برمجية وإذا لم يتحقق يتم تنفيذ جملة برمجية أخرى ثم بعدها ينفذ باقي جمل البرنامج.



25

اكتب برنامج يطبع عبارة x is positive في حال ادخلت قيمة موجبة للمتغير x،  
وعبارة x is negative في حالة تم ادخال قيمة سالبة ؟

```

import java.util.Scanner;
public class Posandneg {

 public static void main(String[] args) {
 Scanner input = new Scanner(System.in);
 String z = input.nextLine();
 int x = input.nextInt();
 if (x > 0)
 System.out.println(x + "is positave");
 else
 System.out.println(x + "is negative");
 }
}

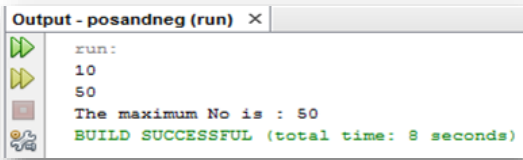
```

26

اكتب برنامج يقوم بتحديد أكبر عدد بين عددين صحيحين ؟

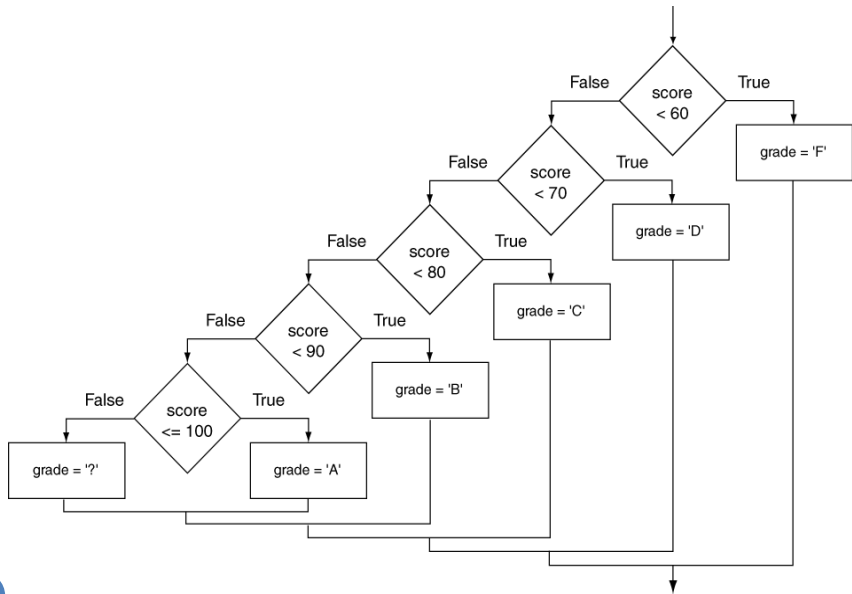
```
import java.util.Scanner;
public class Posandneg {

 public static void main(String[] args) {
 Scanner input = new Scanner(System.in);
 int a =input.nextInt();
 int b =input.nextInt();
 if (a>b)
 System.out.println("The maximum No is :"+a);
 else
 System.out.println("The maximum No is : "+b);
 }
}
```



27

if-else-if Flowchart

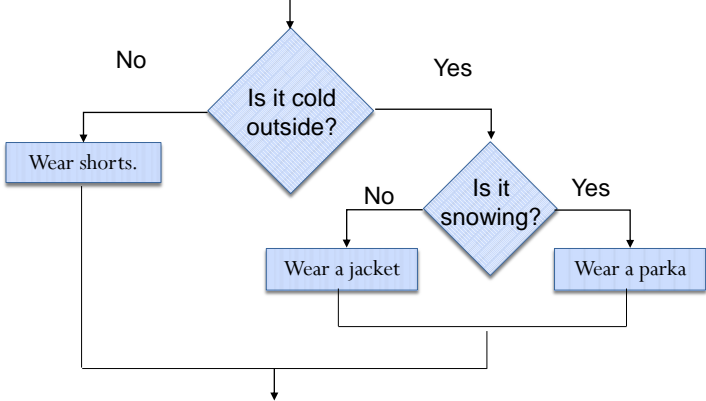


28

### جمل If المتداخلة Nested if Statements

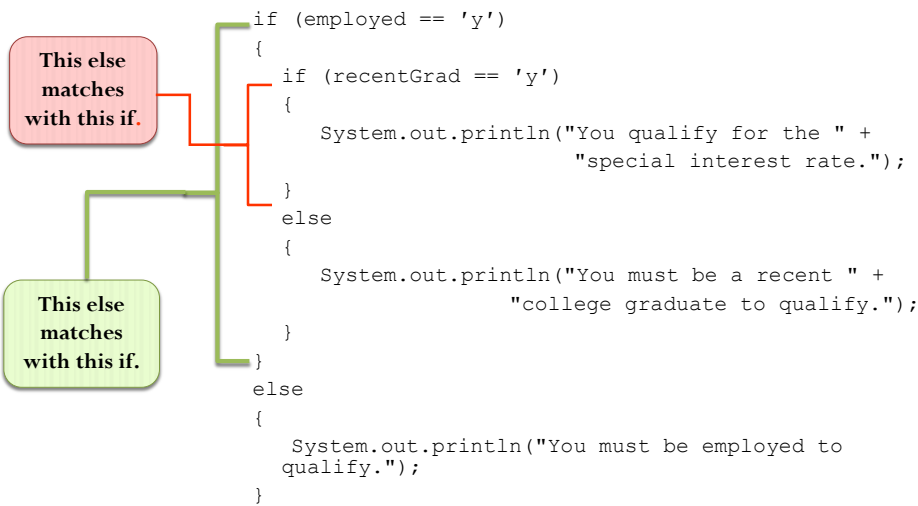
- إذا ظهرت جملة if داخل جملة if أخرى (سواء كانت single أو block) تسمى جمل if متداخلة *nested if*.
- If الداخلية تنفذ فقط في حال أن كان ناتج if الخارجية true.

### Nested if Statement Flowcharts



29

### if-else Matching



30

## المؤثرات المنطقية Logical Operators

- لغة جافا توفر معاملين منطقيين *binary logical operators* هما ( & , | )  
الليذان يستخدمان للمقارنة بين التعبيرات المنطقية *boolean expressions*.
- كما توفر المعامل المنطقي ( ! ) ليعطي عكس حقيقة التعبير المنطقي (لنفي).

| المؤثر | المعنى | التأثير                                                                                                                                                      |
|--------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| &&     | AND    | يربط تعبيرين منطقيين ( two boolean expressions ) لينتج تعبير منطقي واحد. ويكون التعبير الناتج true إذا وإذا فقط كان كل من التعبيرين true. وإلا فسينتج false. |
|        | OR     | يربط تعبيرين منطقيين لينتج تعبير منطقي واحد. ولكي يكون التعبير الناتج true يجب أن يكون أحد التعبيرين أو كلاهما true. وإلا فسينتج false.                      |
| !      | NOT    | المؤثر ! يعكس حقيقة التعبير منطقي. إذا تم تطبيقه على تعبير false، فإنه سيُرجع true أما إذا تم تطبيقه على تعبير true، فإنه سيُرجع false.                      |

31

## The && Operator

- المعامل المنطقي AND يُعبر عنه باستخدام &&.
- عند استخدامه مع تعبيرين منطقيين فإن الناتج يكون true إذا وإذا فقط كان كلا التعبيرين true. وإلا فسينتج false. كما يلي:

| Expression 1 | Expression 2 | Expression1 && Expression2 |
|--------------|--------------|----------------------------|
| true         | false        | false                      |
| false        | true         | false                      |
| false        | false        | false                      |
| true         | true         | true                       |

32



## The || Operator

- المعامل المنطقي OR يُعبر عنه باستخدام `||`.
- عند استخدامه مع تعبيرين منطقيين فإن الناتج يكون `false` إذا وإذا فقط كان كلا التعبيرين `false`. كما يلي:

| Expression 1 | Expression 2 | Expression1    Expression2 |
|--------------|--------------|----------------------------|
| true         | false        | true                       |
| false        | true         | true                       |
| false        | false        | false                      |
| true         | true         | true                       |

33

## The ! Operator

- العملية المنطقية NOT (نفي) يعبر عنه باستخدام المعامل `!`.
- إذا كان التعبير غير صحيح، فإن نفيه سيكون غير صحيح.
- If an *expression* is true, *! expression* will be false.

مثال:

```
if (!(temperature > 100))
 System.out.println("Below the maximum temperature.");
```

يعني: إذا كانت قيمة المتغير `temperature` أكبر من 100 فعلاً أي صحيحة سيتم نفيها ليكون الناتج `False` ولن تُطبع الجملة `Below the maximum temperature` ولكي تُطبع تلك الجملة يجب أن تكون قيمة المتغير `temperature` أقل من 100

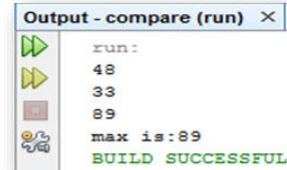
| Expression 1 | !Expression1 |
|--------------|--------------|
| true         | false        |
| false        | true         |

34

## Example 1:

- اكتب برنامج يعمل على إيجاد أكبر قيمة بين ثلاثة أرقام باستخدام أدوات المقارنة فقط ( الشرط البسيط )

```
import java.util.Scanner;
public class Compare {
 public static void main(String[] args) {
 int max = 0;
 Scanner in = new Scanner (System.in);
 int x = in.nextInt();
 int y = in .nextInt();
 int z = in.nextInt();
 max = x;
 if(y>max)
 max = y;
 if (z>max)
 max = z;
 System.out.println("max is:"+max);
 }
}
```



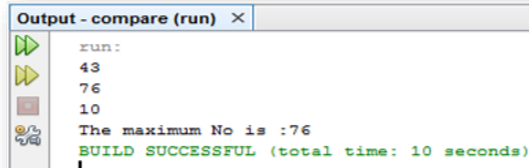
```
Output - compare (run) X
run:
48
33
89
max is:89
BUILD SUCCESSFUL
```

35

## Example 2:

- اكتب برنامج يعمل على إيجاد أكبر قيمة من ثلاثة أرقام باستخدام أدوات المقارنة والأدوات المنطقية ( شرط مركب )

```
2 import java.util.Scanner;
3 public class Compare {
4 public static void main(String[] args) {
5 Scanner in = new Scanner (System.in);
6 int x = in.nextInt();
7 int y = in .nextInt();
8 int z = in.nextInt();
9 if(x>y && x>z)
10 System.out.println("The maximum No is :"+x);
11 else if (y>x && y>z)
12 System.out.println("The maximum No is :"+y);
13 else
14 System.out.println("The maximum No is :"+z);
15 }
16 }
```



```
Output - compare (run) X
run:
43
76
10
The maximum No is :76
BUILD SUCCESSFUL (total time: 10 seconds)
```

36

## جملة الاختيار Switch

- تعتبر جملة switch من جمل التحكم في سير البرنامج ويمكن استخدامها عوضاً عن الكثير من جمل if التي تعتبر جملة لخيار أو خيارين (true/false) بينما switch للخيارات المتعددة.
- وعدد الخيارات Case غير محدد بعدد معين بل حسب الحاجة.

37

## الشكل العام لجملة الاختيار Switch

```
switch (testExpression)
{
 case Value_1:
 statement(1); // جملة (أو مجموعة جمل) برمجية
 break;
 case Value_2:
 statements(2); // جملة (أو مجموعة جمل) برمجية
 break;
 :
 :
 // case statements may be repeated
 //as many times as necessary
 default:
 default statement // جملة (أو مجموعة جمل) برمجية
}
```

38

- Switch: كلمة محجوزة لتحويل مسار التحكم إلى احد الخيارات المحصورة بين قوسي المجموعة، حسب قيمة التعبير testExpression.

- testExpression: التعبير الذي يتم اختباره وحسب قيمته يتم تنفيذ اجراء معين.

```
Scanner
double
switch(grade) {
 //
```

incompatible types; possible lossy conversion from double to int  
----  
(Alt-Enter shows hints)

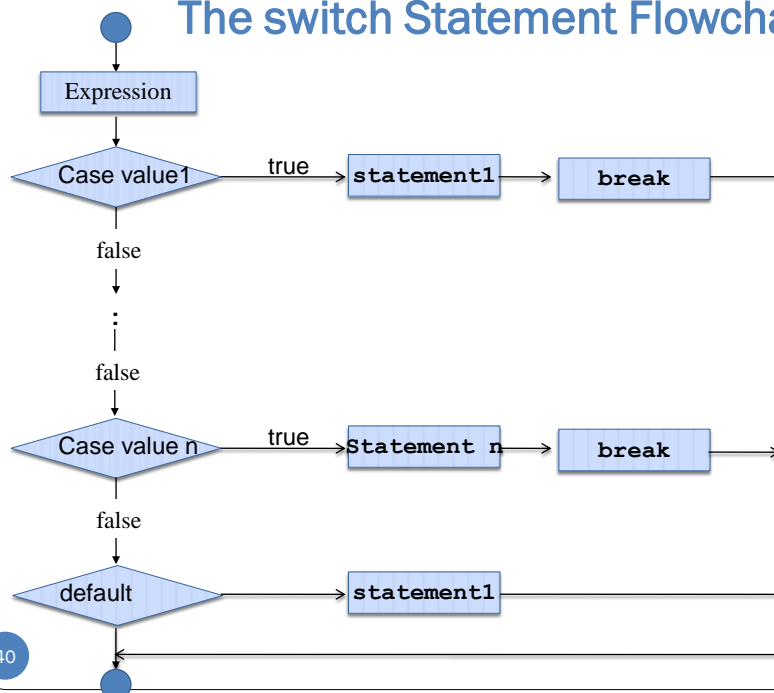
- Case: عنوان الحالة التي يجب أن توافق قيمة التعبير على أن يتبع هذه القيمة شارحة (:).

- break: تستعمل لفصل كل حالة case عن الحالة التي تليها.

- default: وتنفذ عندما تكون قيمة التعبير غير قابلة للتحقق (أي لا توافق أي حالة Case)، وهي اختيارية.

39

## The switch Statement Flowcharts



40

اكتب برنامج يطبع جملة ترحيب معينة حسب قيمة رقمية يدخلها المستخدم.

```
package switch1;
import java.util.Scanner;
public class Switch1 {
 public static void main(String[] args) {
 Scanner input = new Scanner(System.in);
 int a = input.nextInt();
 switch(a)
 {
 case 1:
 System.out.println("Welcom");
 break;
 case 2:
 System.out.println("Haw are you ?");
 break;
 case 3:
 System.out.println("Good Morning");
 break;
 default:
 System.out.println("Good By");
 }
 }
}
```

```
3
Good Morning
BUILD SUCCESSFUL (total time: 6 seconds)
```

41

ماهي نتيجة تنفيذ البرنامج في حالة عدم وجود جملة الـ break

```
1 package switch1;
2 public class Switch1 {
3 public static void main(String[] args) {
4 char ch = 'a';
5 int x=0;
6 switch(ch)
7 {
8 case 'a':
9 x =10;
10 case 'b':
11 x =20;
12 case 'c':
13 x =30;
14 }
15 System.out.println("x="+ x);
16 }
17 }
```

```
run:
x=30
BUILD SUCCESSFUL (total time: 0 seconds)
```

42

## Example 5:

- ماهي نتيجة تنفيذ البرنامج في حالة وجود جملة الـ break ؟

```

1 package switch1;
2 public class Switch1 {
3 public static void main(String[] args) {
4 char ch = 'a';
5 int x=0;
6 switch(ch)
7 {
8 case 'a':
9 x =10;
10 break;
11 case 'b':
12 x =20;
13 break;
14 case 'c':
15 x =30;
16 break;
17 }
18 System.out.println("x="+ x);
19 }
20 }

```

run:  
x=10  
BUILD SUCCESSFUL (total time: 0 seconds)

43

## Example 6:

```

2 import java.util.Scanner;
3 public class Selectswitch2 {
4 public static void main(String[] args) {
5 Scanner input =new Scanner(System.in);
6 String grade=input.nextLine();
7 switch(grade){
8 case"pass":
9 System.out.println(" الطالب ناجح ");
10 break;
11 case "fail":
12 System.out.println(" الطالب راسب ");
13 break;
14 }
15 }
16 }

```

Output - selectswitch2 (run) ×

```

run:
pass
الطالب ناجح
BUILD SUCCESSFUL (total time: 6 seconds)

```

44

## ملاحظات

- في حال اعدنا كتابة نفس الحالة Case مرة اخرى فلن يقبلها

```
switch(grade)
{
 case 3:
 System.out.print("B");
 duplicate case label

 "B";
 (Alt-Enter shows hints)
 case 3:
 System.out.print("B");
}
```

- في حالة عدم وجود break في احد الحالات فإن سيزحف على الحالة التالية كما في المثال:

```
int grade=in.nextInt();
switch(grade)
{
 case 4:
 System.out.println("A");
 case 3:
 System.out.println("B");
 break;
 case 2:
 System.out.println("C");
 break;
 case 1:
 System.out.println("F");
 break;
}
```

run:  
4  
A  
B

45

غالباً لا نعطيها قيمة كما في هذا  
المثال، بل نترك للمستخدم حرية  
الادخال (القراءة).

```
19 char grade='A';
20
21 switch(grade)
22 {
23 case 'A':
24 System.out.println("4");
25 break;
26 case 'B':
27 System.out.println("3");
28 break;
29 case 'C':
30 System.out.println("2");
31 break;
32 default: System.out.println("Error");
33 }
34
```

selectionStatements.SelectionStatements > main > switch (grade) > case &apos;B&apos;

Output - selectionStatements (run)

```
run:
4
BUILD SUCCESSFUL (total time: 0 seconds)
```

46

```

int grade=in.nextInt();
switch (grade)
{
 case 4:
 System.out.println("A");
 case 3:
 System.out.println("B");
 break;
 case 2:
 System.out.println("C");
 break;
 case 1:
 System.out.println("F");
 break;
 default: System.out.println("خطأ في الإدخال");
}

```

selectionStatements.SelectionStatements > main > switch (grade) > default: >

ut - selectionStatements (run) ⌘

run:  
44  
خطأ في الإدخال

47

```

Scanner in= new Scanner(System.in);
int month= in.nextInt();
int numberOfDays = 0;
switch(month){
 case :1
 case :3
 case :5
 case :7
 case :8
 case :10
 case :12
 numberOfDays=31
 break;
 case :4
 case :6
 case :9
 case :11
 numberOfDays=30
 break;
}

```

48



```
case :2
 System.out.println("Plz. enter the Year:")
 int year= in.nextInt()
 if (year%4==0)
 numberOfDays=29
 else
 numberOfDays=28
 break
}
```

49

## شكرًا لإنصاتكم



السؤال مفتاح للعلوم كلها...  
تذكر أن نيوتن عندما سأل لماذا وقعت التفاحة وصل إلى قوانين الجاذبية...